

第5章 宇宙機器に学ぶ安全なソフトウェアの作り方

安全意識と誇りを持って設計・検証に取り組む

片平真史
石濱直樹

ここでは，宇宙分野における信頼性向上や安全性確保のための考え方と，宇宙航空研究開発機構（JAXA）で実施している安全のための取り組みについて解説する．（編集部）

宇宙分野にも幅広い種類のソフトウェアがあります．例えば，人工衛星（写真1）に搭載されるソフトウェアは，規模や機能の面で家電製品などに組み込まれるソフトウェアと似ています．そして，開発の課程において抱えている問題点も，ほかの組み込みソフトウェアとよく似ています．

一方，宇宙分野のソフトウェアは以前から高い信頼性や安全性を要求されてきており，その点でいろいろと工夫を重ねています．ここでは，宇宙分野において取り組んできた手法や考え方をベースに，ソフトウェアの安全性を高める方法について解説します．

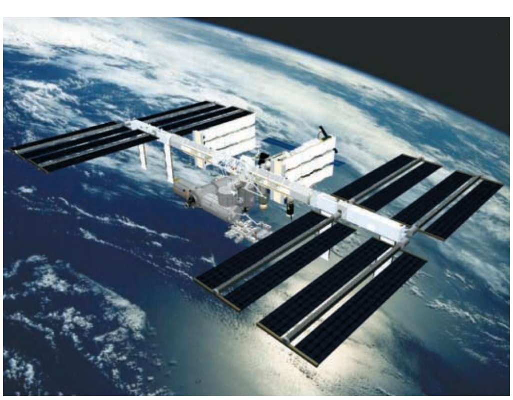


写真1　国際宇宙ステーション（完成イメージ図）

1．信頼性向上と安全性確保の違いを理解する

人工衛星などの宇宙機器は，宇宙という特殊環境で使用する（例えば，宇宙放射線の影響により，コンピュータのデータにビット化けが発生する）ため，宇宙用に開発された耐放射線性能の高いCPUを搭載しています．ただし，宇宙用CPUを使用したとしても，ソフトウェアにおけるビット化けなどの影響を考慮して設計する必要があります．

ソフトウェアで故障耐性を確保するための手法として，Nバージョン・プログラミング（同じ処理を異なるアルゴリズムで複数実装しておき，並列処理して計算結果を多数決する）注1というものがあります．この有効性を報告する論文もある一方，開発コストの増加や，多数決で計算結果を選択することによって一貫性がなくなり，新たな問題を引き起こす可能性がある，などと指摘されています(1)(2)．

スペース・シャトルの設計においては，Nバージョン・プログラミングのほかに，それらを監視して異常を検知する役割を持ったコンピュータとソフトウェアを用意するなどの方法も採られています．

また現在は，安全にかかわる複数の制御系や監視系を，1台のコンピュータに搭載されたソフトウェアで処理する場合があります（右掲のコラム「宇宙ステーション時代になって認められたソフトウェアの冗長性」を参照）．このよ

注1：IEC 61508で紹介されているdiverse programmingの一つの技法を指す．

KeyWord　Nバージョン・プログラミング，高信頼性検証，FTA，IV&V，SPiCE for Space，TOPPERS/HRP，安全確保

うな状況では，リアルタイムOSにおける安全性確保が重要となります．安全解析や安全設計のためにOSの内部詳細情報を分析する必要があるため，宇宙航空研究開発機構(JAXA)では，リアルタイムOSの高信頼性検証プロセスを独自に策定するとともに，オープン・ソース・ソフトウェアであるTOPPERS OSをベースとして安全機能を強化したリアルタイムOSを開発しています(p.58のコラム「リアルタイムOSの信頼性を検証する」を参照)．

●「信頼性が高い＝安全」ではない

製品の信頼性が向上することは，出荷後のリスクを低減するために大変重要です．また，製品(ソフトウェア)をテストすることにより，製品の信頼性を向上できることも自明の理です．ところで，製品の安全に対しては，信頼性を向上するだけで十分なのでしょうか．

意図的に事故を起こすような製品を作る人はいません．準備された機能が想定外の動きをしたときや必要な機能が欠落していたときに，それが大きな事故につながります．従って，要求された機能が想定された条件のもとに動作するだけではだめなのです．事実，1996年に起きた，ESA(European Space Agency；欧州宇宙機関)のアリアンV型ロケットの事故(異常検知により自動爆破した)は，設定された機能は求められた通りに動作しましたが事故に至りました[注2]．

製品そのものにどのような潜在的危険があり，その製品に組み込まれたソフトウェアがどのように影響する可能性があるのかを，安全性という観点からしっかりと分析する必要があります．

宇宙分野においては，以下のような安全性の基本方針があります．

- どのような故障が二つ重なって発生しても，人を殺さない．製品を喪失しない

A. 直接原因となる制御系
機能そのものが危険な事象に影響する機能
ケース1）機能不動作が人を殺す，負傷させる
- 必要なときに動作しない
- 動作中に不意に止まる
- 停止時に想定外の止まり方をする

ケース2）機能動作が人を殺す，負傷させる
- 動いてはいけないときに動き出す
- 停止しなければならないときに止まらない
- 動き出すときに想定外の動きをする

B. 安全監視系
危険な事象を監視・検知し，抑制する機能

図1 ソフトウェアが関係する安全機能
ソフトウェアは製品に組み込まれることにより，危害を及ぼす可能性が発生する．特に，制御系や安全監視系の機能を担ったときに，安全にかかわってくる．

- どのような故障が(単独で)発生しても，人を負傷させない．製品の重要な機能を喪失させない

ソフトウェア安全性の規格も，この基本方針に従って整備されています．これらの前提の下，ソフトウェアの特性を考慮して，安全確保に取り組んでいます．

あえて言うまでもありませんが，ソフトウェアだけでは人を殺したりはしません．ソフトウェアがシステム(製品)に組み込まれて初めて，その危険性が発生します．では，ソフトウェアがどのような形で製品全体の安全性に影響するのでしょうか．具体的事象は，業種や製品によってまちまちですが，一般的には図1に示す機能について，安全を考慮する必要があると考えられます．

なお，宇宙ステーション計画における安全要求「SSP50038(宇宙ステーションのコンピュータ制御システムに対する安全要求)」[(3)]にも，図1のA(直接原因となる制御系)に相当する項目があります．Must Work Function(ケース1)とMust Not Work Function(ケース2)を識別し，安全要求を適用するように規定されています．

注2：当初の想定と異なる利用環境(再利用)で動作したため，事故に至った．もちろん，再利用時の検証不足も問題として指摘されるべきだ．

COLUMN 宇宙ステーション時代になって認められたソフトウェアの冗長性

スペース・シャトルを設計していた時代(1970年代後半～1980年代後半)には，1台のコンピュータ上のソフトウェア機能として二つ以上の冗長機能を実装することは，安全設計上，許容されていませんでした．しかし，コンピュータの能力向上などにより，宇宙ステーションを設計する時代(1980年代後半～現在)になってからは，1台のコンピュータ上のソフトウェアで二つ以上の制御系や監視系の処理を行うことが許容されるようになっています．

この際に重要なのは，ソフトウェアの処理の独立性(リアルタイムOSレベルまで含む)を確保することです．いかなる欠陥が発生しても，同時に二つ以上の処理に影響しないことを保証する必要があります．

直接原因となる制御系（図1のA）のうち，ケース1は，自動車におけるブレーキのようなものです．その機能が動作しないことにより，直接的に危険な事象に影響するものが該当します．一方，ケース2は，エレベータの制御などが該当します．人の乗り降りの最中に不用意に動作してしまうようなケースです．通常は，ケース1とケース2の両方に該当する機能が多く識別されることになります（例えば，エア・バッグなど）．

また，安全監視系（図1のB）は，火災時の自動消火機能などのようなものです．その機能が故障することによって，直接的に火災が発生するわけではありませんが，別の原因で発生した火災を監視・検知し，延焼を防止し，鎮火する機能がこれに該当します．

図1に示したような潜在危険に関連する機能をリストアップすれば，ソフトウェアが製品の安全性に影響を及ぼす事象を洗い出すための一つのヒントとなります．

2．ソフトウェアの安全評価プロセス

宇宙ステーション計画などにおいては，ソフトウェアの安全性はシステムの安全性と同じように詳細に分析・審査されます．ここではそのプロセスを参考に，ソフトウェア安全評価を行うためのステップを紹介します．

● ステップ1：安全の視点で製品を見つめる

●製品にとって危険な事象にはどのようなものがあるか？

まず，最初の段階として，製品（システム）全体として，どのようなことが潜在危険として危ぶまれるかをじっくり洗い出すことになります．この段階では，さまざま立場の人が参加して抽出作業をするとよいでしょう．

●危険な事象に関連するソフトウェア機能はどのようなものがあるか？

ここでは，ソフトウェアがどのように製品の安全性に影響を及ぼすかを分析します．図1に示した観点とともに，システムに関するフォールト・ツリー分析（システムFTA）やソフトウェアに関するフォールト・ツリー分析（ソフトウェアFTA），故障モード影響解析（FMEA；failure mode & effects analysis）などがここで役に立ちます．例として，「宇宙機を失ってしまう」という危険な事象に対して，ソフトウェアがどのように影響するのかを分析したFTAを図2に示します．

まず初めに，システムFTAのトップ事象「宇宙機喪失」の原因となる子事象を識別し，さらにその子事象という形で段階的に要因を詳細化します．次に，このシステムFTAの分析から事象を連結する形で別途ソフトウェアFTAを進め，ソフトウェア内部の要因を詳細化していきます．

ソフトウェアFTAにおいては，ソフトウェア内の機能故障を想定するのではなく，与えられた機能に対して，外部要因やタイミングなどの問題により，機能が不用意に動作する/動作しないという切り口で分析すると，うまく分析できます．

ソフトウェア要因としては，できる限り詳細に子事象を識別することが望ましいのですが，ある程度の詳細レベル（モジュール分解レベル）まで達成したところで，それ以上の詳細化は困難になります．そこでストップです．

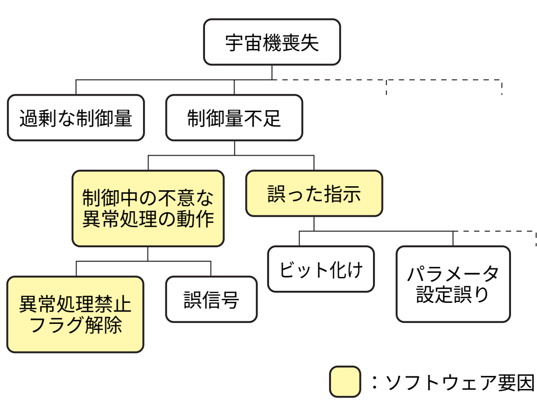


図2　システムに関するフォールト・ツリー分析（システムFTA）
システムFTAでトップ事象から子事象を識別していく．そして，分析した子事象（ソフトウェア要因）をさらに分析するために，ソフトウェアFTAを実施する．

● ステップ2：安全性を設計する

ソフトウェアFTAの末端事象に対して，どのようにソフトウェア要因を制御するのかという検討が始まります．ソフトウェア・インヒビット（潜在危険に至る指示の実行を不可能にするためのソフトウェア機能），故障許容設計，安全マージン設計など，さまざまな設計解が存在しますが，システム構成やリソースなどを考慮した上で設計解を決定し，上記FTAに反映することになります．

ここで重要なのは，無理にソフトウェアですべて解決しようとしないことです．ソフトウェアに固執することでソフトウェアをさらに複雑なものにしてしまい，かえって危険性を増すことになります．また，検証がさらに困難に

なったり，開発コストが増加することにもなりかねません．ソフトウェアだけではなく，コンピュータを含むハードウェアによる設計解を考えたり，機能安全による設計解だけではなく，いわゆるpassive safetyによる解決策を含め，幅広い視野を持って検討することがとても重要です．

● ステップ3：安全性を検証する

安全設計の結果に基づいて，設計解が製品に確実に組み込まれ，正しく動作することを検証します．そのために必要な検証プロセスを計画し，漏れのないように検証を実施します．検証結果を記録して，安全を立証します．

3．安全性確保に有効な取り組み

JAXAではこれまで述べてきたような安全評価の取り組みのほか，安全確保のために，ソフトウェア独立検証・妥当性確認（IV&V：Independent Verification & Validation）やプロセス改善にも取り組んでいます．

● コストに合わせて検証・妥当性確認を実施

IV&Vとは，開発組織から独立した組織が，独自の観点で検証と妥当性確認（V&V）を実施することを言います．宇宙分野においては，スペース・シャトル「チャレンジャー」の事故をきっかけに，ソフトウェアIV&Vが強化されるようになりました．米国航空宇宙局（NASA）は1993年にIV&V専門の施設を設立し，NASAの全プロジェクトを対象に活動を開始しました[注3]．JAXAにおいても，宇宙ステーション計画をきっかけに1996年よりIV&V活動を継続的に実施しています．

宇宙分野においても，ソフトウェアの検証コストは厳しく制限されています．JAXAは，それぞれのソフトウェアに合わせて，コスト対効果の高いIV&V活動を目指しています．具体的には，各検証技術に対しての効果（重大バグの早期抽出）を計測して，プロジェクトの特徴やコストなどの制約事項を考慮した上で検証技術を選定・最適化するためのモデルを構築しています．

図3は，IV&Vとして実施するべき評価観点（縦軸）に対して有効なIV&V技術を，開発フェーズ（横軸）に沿って，実施コスト（奥行き軸）とともに例示したものです．例えば，人命に影響するような重要な安全関連機能では，仕様の完全性を検証するために「モデル検査」が有効です．一方，検証コストが少なく，何かあっても一時的にデータを損失する程度の影響しか及ぼさない機能については，過去の経験則をまとめた「チェック・リスト」によるレビューに作業を限定する方法があります．

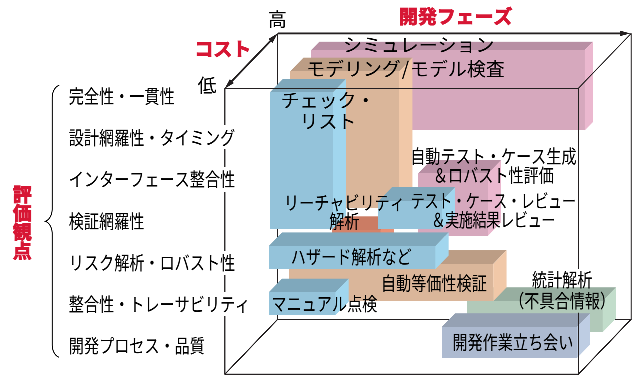


図3 IV&Vの選択モデル
評価観点と開発フェーズ，実施コストごとに，合致する検証技術を図示している．

このように，対象機能の安全上の重要性と検証コスト制約に合った技術を選択し，最適化することにより，限られたコストの中でも安全性の向上に貢献することができます．現在筆者らは，NASAおよび欧州宇宙機関（ESA）との国際協力の下，ソフトウェアIV&Vとして有効な技術の体系化と標準化を行っています．

● プロセス改善とメトリクス

プロセス改善活動としては，事故や重大な不具合を分析し，プロセス上の課題を識別しています．米国の航空機事故調査で用いられるような事故モデル分析技術[4]を利用して，改善が必要なプロセスを抽出し，最も改善効果の上がるプロセスから改善を実施しています[注4]．

また，宇宙分野の場合，開発組織はJAXA以外に開発メーカや下請け企業などがあり，それらの企業の協力によって開発を進めています．そのため，開発プロセスにおける各企業の役割やプロセスのインターフェースを明確に規定する必要があります（統合プロセス）．信頼性向上と安全性確保の観点からプロセス・メトリクスを導入し，プロセスの良否を可視化しています．

関連メーカごとに取得できるメトリクス・データの種類

注3：NASAのIV&Vについては，http://www.ivv.nasa.gov/を参照のこと．
注4：通常は，一つの事故や不具合情報から複数の改善するべきプロセスが識別でき，どのプロセスを改善しても事故や不具合の発生を抑制できることが分かる．

が異なるため，JAXAでは評価関数を用意してデータを加工し，全体を定量的に評価できるように工夫しています．

プロセス・アセスメントの観点では，欧州を中心として宇宙分野特有のベスト・プラクティスを考慮したプロセス・モデル「SPiCE for Space」が整備されています．ESAは，既にこのプロセス・モデルに基づいたアセスメントを開始しています．

● 安全は3段階で確保する

安全なシステム（ソフトウェア）の開発を考える場合，まず，安定した開発プロセスにより製品の品質を向上させることが不可欠です．また，テストにおいて，異常故障試験，負荷試験，ロバスト試験などによるテスト・ケースを充実することにより，システムの信頼性を向上することも当然重要となります．このような，可能な限り欠陥を排除する（火種を取り除く）活動によって「欠陥ゼロ」を目指す活動

COLUMN リアルタイムOSの信頼性を検証する

宇宙機器に搭載するソフトウェアの開発において，これまでは，「リアルタイムOSは動いて当たり前」という認識が前提にあり，内部解析を十分に行うことはほとんどありませんでした．その結果，リアルタイムOSの不具合により開発に影響をきたすことが増えてきました．また，CPUの高速化に伴い，搭載アプリケーション・ソフトウェアに高度で複雑な機能が要求されるようになってきており，リアルタイムOSは今まで以上に重要な役割を担っています．そのような状態でリアルタイムOSを十分に検証しないまま使い続けることは，信頼性・安全性の観点からリスクがあります．

そこで筆者らは，高信頼性や安全性が求められるシステムに使われるリアルタイムOSを検証するためのプロセスを定義しました．また，TOPPERSプロジェクトと共同で，μITRON 4.0仕様のスタンダード・プロファイルをベースに，信頼性を高める独自の機能を盛り込んだリアルタイムOS「TOPPERS/HRP」を開発し，宇宙用のCPUに実装しました．

● 各業界の標準から検証要求を抽出

ソフトウェア全般に対する安全基準やガイドラインとしては「DO-178B」や「IEC 61508」などがありますが，リアルタイムOSそのものの安全を考えるためのガイドラインのようなものは，世の中に存在しません．また，これらの安全基準には「何をしなければならないか（what）」については記述されていますが，「どのように実施しなければならないか（how）」については記述されていません．そこで筆者らは，リアルタイムOS単体に対して，特に検証に着目し，どのような検証をどのように実施すればリアルタイムOSとしての品質を確保することができるかを検討しました．

医療機器，原子力発電所，鉄道，軍事機器，民間航空機という五つの業界における代表的な12の標準をベースとして，必要な検証要求を抽出しました（図A）．これらの項目を検討し，最終的に，検証プロセスの一部として定義しました．

● 処理時間予測や保護機能によって信頼性を高める

TOPPERS/HRPの開発にあたっては，システムを安全に運用できるように，処理を確実にリアルタイムに実行することを重視しました．応答時間を一定の範囲内にしたほか，必要な処理時間を予測する機能や，複数の処理要求が同時に発生した場合でも目的の時間内に完了させるための機構・機能などを実装しました．また，リアルタイム性以外の機能として，主に以下の3点について機能仕様を作成し，開発を進めました．

- 搭載ソフトウェアにおいて過去に起こった問題を防ぐために必要な機能（アプリケーション・ソフトウェアが暴走してもメモリ上の使用禁止領域にアクセスできない保護機能など）
- システム全体として信頼性を上げるために必要な機能（リアルタイムOSやアプリケーション・ソフトウェアの動作状況を記録するログ機能など）
- 個々に設計された機器を統一的に扱える機能（機器の起動時の処理や機器ごとの処理の流れを統一的に管理する機能など）

図A リアルタイムOS検証要求の抽出例

五つの業界標準や，全体を網羅した機能安全規格（IEC 61508）などを参照し，要求されている検証項目を洗い出した．

国際規格		規格名	RTCA DO-178B	JIS C 0508-7 (IEC 61508)	IEC 62279	IEC 880 (IEC 60880)	DEF STAN 00-42 (PART 2)	General Principles of Software Validation
		分野	民間航空機	産業一般	鉄道	原子力発電	軍事機器	医療機器
要求事項テスト	内部要求事項テスト		○	—	—	○	○	—
	外部要求事項テスト		○	—	—		—	—
入力域のテスト	正常入力域のテスト		○	—	—	—	—	○
	代表値のテスト/同値分割		○	○	○	○	—	—
	境界値分析 /異常入力域テスト /ロバストネス・テスト /極端な値のテスト /クラッシュ・テスト		○	○	○	○	○	○

が，安全確保の点から避けることのできない重要な第1歩となります(**図4**の①)．

しかし現実的には，どのような努力をしても完全に欠陥をゼロにすることは不可能です．そこで「欠陥があってもシステムとして表面化しない仕組み」を作り込むことが必要になります(**図4**の②)．多様性や機能冗長化などのさまざまな技術がありますが，ここで過剰設計にならないように注意が必要です．先ほども述べたように，過剰設計は複雑化を招き，安全性低下の原因になります．なお，宇宙機ソフトウェアの事例においては，この②の段階における対策が機能分量として大半を占めています．

万が一，欠陥が表面化してしまった場合のためには，「表面化しても抑制(低減)できる仕組み」を準備し，危険事象を検知して抑制することが必要となります(**図4**の③)．宇宙分野では，すべての危険事象を検知する仕組みが要求されており，これが最後のとりでとなっています．

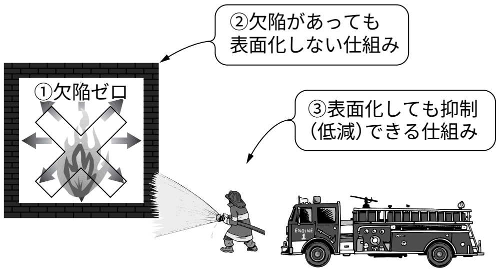


図4　安全確保の考え方

火を出さないようにし(欠陥ゼロ)，たとえ火が出ても防火壁で防ぎ(表面化しない仕組み)，表面化しても消火する(被害を抑制できる仕組み)．この3段階で安全を確保する．

● 日常の意識が安全なソフトウェアを作る

安全確保の活動は大変地道な，気力のいる作業です．それなのに「安全に動作して当たり前」と思われてしまい，その努力が理解されることは非常にまれです．また，事故として表面化したとたん，やり玉に挙げられてしまい，急に「安全」という言葉が世間に踊るようになります．製品を作る現場の担当者としてはやり切れないものですが，宇宙分野，特にNASAでは，安全文化の一端を担う技術者はその作業に大変誇りを持っています．また，周囲からも「安全が設計を決める」と高く評価されています．

安全文化の崩壊を回避するための一つの方策として，IEC 61508などの認証によって安全文化を構築することは大変有意義だと思いますが，それも関係者の日常の意識(モラル)が維持されてこそ，効果が上がります．

最近の国内における事故事例からも分かるように，どんなに素晴らしい認証活動や設計技術，解析技術があったとしても，すべては企業およびその開発者・運用者のモラルに依存してしまいます．「セーフティ・クリティカル・システムを開発している」という誇りが，事故回避に大きく貢献すると思います．

また，認証に安住してしまうと大変危険です．常に「この製品は本当に安全か？」と問い続け，安全のプロフェッショナルとしての姿勢を維持し続ける必要があります．Nancy Leveson氏は，安全評価プロセスに費やされるコストや技術者の意識と事故の発生状況の関連性を分析した結果，「どのような優れた安全文化が構築できても，事故は，安全文化の形骸化や心のすき間に忍び寄る(長い間事故が起きないうちに少しずつ安全の重要性を忘れていき，注意を払わなくなる)」というデータを示しています．

私たち技術者は，このような状況を真剣に受け止め，誇りを持って「安全」な製品作りを進めていきたいものです．

5

参考・引用*文献

(1) John Knight and Nancy Leveson ; An Experimental Evaluation of the Assumption of Independence in Multi-Version Programming. IEEE Transactions on Software Engineering, vol. SE-12, no. 1, pp. 96-109A, Jan. 1986.
(2) John Knight and Nancy Leveson ; Reply to the Criticisms of the Knight and Leveson Experiment, ACM Software Engineering Notes, Jan. 1990.
(3) Computer-Based Control System Safety Requirements, SSP 50038 Rev. B, Nov. 1995.
(4) Nancy Leveson ; SAFEWARE : System Safety and Computers, Addison Wesley, 1995.

かたひら・まさふみ
いしはま・なおき
宇宙航空研究開発機構(JAXA) 情報・計算工学センター

<筆者プロフィール>

片平真史．学生時代の副業がきっかけで，ソフトウェア安全や信頼性について研究を始める．Nancy Leveson教授のもとで研究者として働き，現在に至る．小学生時代に某名門少年少女合唱団で培った歌声は今も健在．

石濱直樹．入社以来ソフトウェアの独立検証に携わり，現在は，アプリケーションだけではなくリアルタイムOSを含むソフトウェア全体の信頼性を上げるための技術について研究中．休みの日は，汗水たらして日曜大工に励んでいる．